TOWARDS AUTOMATIC ROBOT DESIGN: A BLACK-BOX FUNCTION OPTIMIZATION APPROACH

A PREPRINT

Zhiyang Xiang
School of Information Science and Engineering, Jishou University
Hunan Province, China, 416000
sbxzy@foxmail.com
z_xiang@hnu.edu.cn

May 5, 2019

ABSTRACT

Robotics is a research topic among the most funded around the world and robots have the potential to largely reduce human labors. The design of robots is, on the other hand mental labor intensive. It is surprising that automatic robot design, as an important research topic, remains unexplored till today. In this paper, a automatic design framework is proposed, where the robotics design problem is model as a black-box function optimization process. Given robots parts and a specific purpose that the robot is designed to accomplish, the framework tries to auto-assemble the parts in a way that is best for the purpose. The framework is a simulation and optimization interactive process, where a physics engine or even real world simulations are employed to score a assembling setting according to its capability to accomplished the given task, and a black-box optimization method is employed to optimize the score. Bayesian hyper-parameter optimization is used to carry out the optimizations, for its advantage of requiring less sampling than other methods. Experiments show that the proposed method is able to find acceptable designs of a two joint system within only 5 simulations.

***K*eywords** Robotics · Automatic design · Bayesian optimization

1 Introduction

It is always desirable for Artificial intelligence (AI) to carry out more work for humans. AI, as its physical counter part such as automated machines, has the potential to largely reduce human labors. It is desirable that AI can automatically create arts for entertainments and industrial purposes. Robotics is a combination of mechanics and computer science, and is one of the most funded research areas in AI around the world. Despite the fact that robot control is a well established area, the automatic robot design problem receives much less attention. In this work, a simulation based automatic design method is proposed. The results can be evaluated in real world situation with 3d printers.

In past works, there are studies that consider the automatic design of controlling logic, i.e. the "software" part of robots [1]. In [2], auto assembling problem is studied, however, this study is for constructing industry where almost all contact of parts are fixed in position, thus it is a different problem from robot construction, where most parts are not fixed.

There are two major problems in automatic robot designs, data representation and optimization efficiency. It is well known that robots can be represented as a tree structure (as in ROS [3]) of individual parts. The tree structure is a discrete data structure and parts have continuous properties. The fusion of discrete and continuous data representations can be performed for the construction of a continuous search space. Because of the fact that evaluations of robot designs are expensive, the search algorithm in the constructed search space should be able to reach optima with a few steps of evaluations.

In this paper, the automatic design of robotics is studied. Instead of designing controlling software, or hardware structures for a specific purpose, the auto assembling procedures for general purpose robots are devised.

2 Methodology

Designs of physical structures in real world have a long history of billions of years. One reason is that the evaluations of the fitness of one design (in this case, a physical structure of a species) is slow. The evolution speed of physical structures is partly dependent on evaluation speed. One evidence is the fruit flies widely used in animal experiments. Due to their high reproduction speed, evolutions which usually take generations can be accomplished in a short time.

In the computer world, the physical structure evolution can be almost infinitely fast because the seemingly unlimited computing power that humanity can command. In this work, the robot design is helped with accelerated physical simulations. The framework is illustrated in Fig. 1. The optimization process generates a vector, which is then translated into a physical robot representation. Then this representation is trained with reinforcement learning according to a specified task. After that, the model is evaluated so that how well it can accomplish the task. And the evaluation score is optimized. The optimizer should have the ability to reach optimal with minimum evaluations, which is similar to active sampling to minimize sample counts. Thus, the framework is a active design procedure.

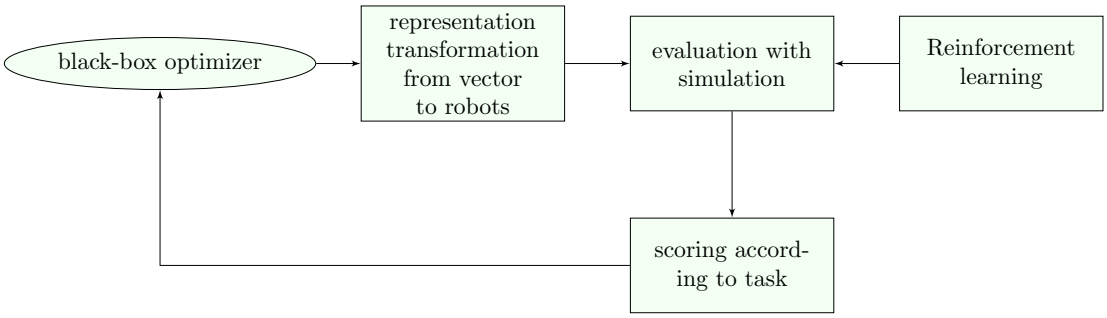


Figure 1: Active design: automatic robot design with simulation and optimization interaction

The input of active design is a set of parts and their specifications S, a reinforcement learning method $\mathcal{L}$, an evaluation function $Score(S, J, \mathcal{L})$ that determines how well the robot can accomplish a specified task while the parts are connected with joints J, and an optimizer

$$\underset{S}{argmax}\, Score(S, \mathcal{L}) \tag{1}$$

Due to the challenges of representation and efficiency as described in the introduction section, two problems need to be addressed in this framework. First, how to change the vector representation, which is the operating objectives of black-box optimizers, to the tree structure recognized by physical simulators or real robot constructors. Second, which optimizer to choose to minimize the evaluations of the $Score$ function.

2.1 Representation

The representation need to be two ways. One is from tree structured data to vector representation, the other is from vector to tree structured parts and their specifications. First, let us see the first transformation, which is to combing a similarities of specification matrix $\mathbf{C}$ and a similarity matrix $\mathbf{G}$ defined by the tree structure. If structure a is different from structure b, despite their similarities of parts specifications, their distance in the similarity space will be larger than c, which has the same tree structure as a.

The fusion of two similarity matrices should preserve $\mathbf{C}$, which is Euclidean distances as well as possible, while following the constraints in $\mathbf{G} = \{g_{ij}\}$ calculated by graph similarities.

Then the problem of fusing two matrices is formalized as

$$\begin{aligned} &\min_{\mathbf{C}'} \sum_i \sum_j (c'_{ij} - c_{ij})^2 \\ &s.t. \forall g_{ij}, g_{mn} \in \mathbf{G}, if\ g_{ij} \le g_{mn}, c'_{ij} \ge c'_{mn} \end{aligned} \tag{2}$$

Because that similarity matrices are symmetric, the upper half of $\mathbf{G}$ is extracted and put in a vector $M = \{m_k\}$ and sort in decreasing order; and add another two vectors $D = \{d_k\}$, $K = \{k_k\}$ for each element $m_k = g_{ij}, d_k = p_{ij}, k_k = c'_{ij}$.

Then, the optimization task of Eq. (2) is transform as

$$\begin{aligned} &\min_{K} \sum_{i} (k_i - d_i)^2 \\ &s.t. \, 0 \leq k_1 \leq k_2 \cdots \leq k_j \cdots \end{aligned} \tag{3}$$

which is equivalent to [4]

$$\begin{aligned} &\min_{K} \frac{1}{2} K^T K - D^T K \\ &s.t. \, 0 \leq k_1 \leq k_2 \cdots \leq k_j \cdots \end{aligned} \tag{4}$$

This is a quadratic programming process. The reverse transformation can be accomplished in two steps. First, find the best structure, then find the specifications. Structure can be found by nearest neighbor in the transformed space, since structure similarity are fully preserved. After that, there is no difficulty in finding the specifications.

2.2 Optimization

Bayesian optimization [5] is chosen to perform the main work of active design, because of its ability to reach optima in a few search steps. This optimization model works under the assumption that parameters to be selected and the score to be maximized form a Gaussian process, i.e. that they form a Gaussian distribution. The process of optimization combined with the simulation steps are given in Algorithm. 1.

Algorithm 1 Active design of robots with Bayesian optimization

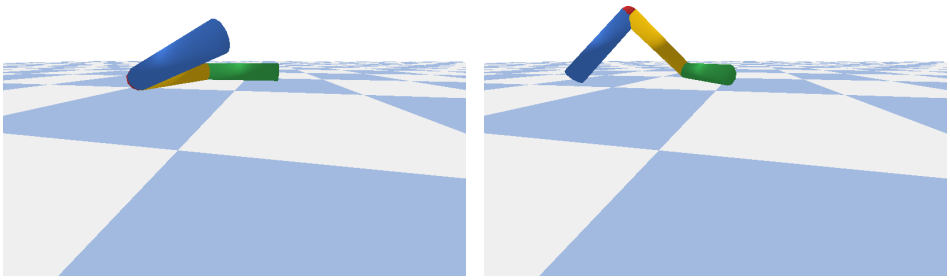
Input: Specifications $\{S\}$, a reinforcement learning method $\mathcal{L}$, an evaluation function $Score(S, J, \mathcal{L})$, maximum evaluations N.
Output: Robot structure and joint specifications.

1: Set GP priors.
2: Generate two space filling J and structure settings, calculate their vector representations with procedures in Section 2.1, and initialize GP with generated vectors and their scores.
3: Add the calculated vectors $v_1 \in V$, $v_2 \in V$ and their scores to a observation set as $O = \{[v_1^T, Score(v_1)]^T, [v_2^T, Score(v_2)]^T\}$.
4: **while** $n \leq N$ **do**
5: Update GP posterior with observed vector representations and their scores in O.
6: From GP, calculate a position $v' \in V$ in the space of robot vector representations that maximizes the GP posteriori.
7: Evaluate $Score(v')$, and add it to O as $[v_n^T, Score(v_n)]^T$.
8: **end while**
9: Output as $\underset{[v_n^T, Score(v_n)]^T \in O}{argmax} \; Score(v_n)$

3 Experiments

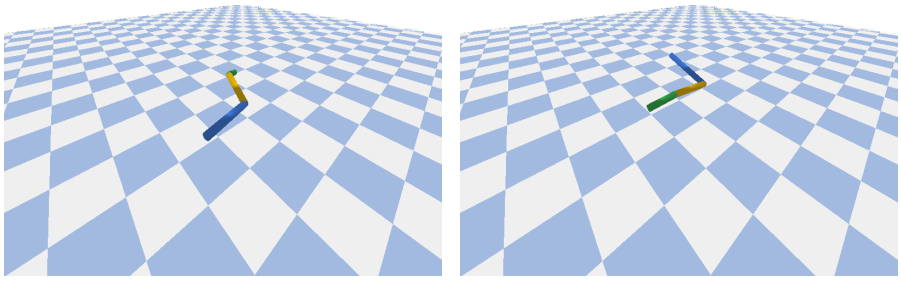
Two toy problems are studied with the ROS and pybullet[6] simulation tools. In the first problem, the input is tree links with the cylinder shape, and joint positions at the end. The task is to "stand up", which means the system will try to lift one of the pieces up, while other two pieces touch the ground. Random initialization with a simple reinforcement learning set up would result in a situation in Fig. 2(a). After 5 evaluations with active design, the robot have achieved the gesture that gives highest score, i.e. that the central link is highest than other gestures as illustrated in Fig. 2(b). The reinforcement method of deterministic policy gradient [7] is selected in experiments. In another problem, the system try to form a gesture of diagonal. Random setup would result in Fig. 3(a), because rotation axises of joints do not allow a diagonal gesture. After 7 steps of active design. The gesture is accomplished as in Fig. 3(b).

Comparisons are carried out with the scipy [8] optimization method Nelder-Mead [9]. The scipy module can achieve similar results to Bayesian optimization, but with more sampling. The results are given in Table 1.



(a) random set up (b) joint optimized with active design

Figure 2: Problem one, optimize joint configurations so that the central link can stay as high as it can be.



(a) random set up (b) joint optimized with active design

Figure 3: Problem two, optimize joint configurations so that a diagonal gesture can be performed.

Table 1: Function evaluation times comparison between Bayesian optimization and scipy's optimization module.

	simulations run	
Optimization method	problem one	problem two
proposed	5	7
scipy	112	128

Each simulation takes about 10 seconds on a mainstream PC. As a result, the proposed method can finish design in minutes, while hours are required using the scipy module. This is because many optimization methods are designed under the assumption that function evaluations are fast, which is not the case in robot simulations or evaluations.

References

[1] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. Automode: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112, 2014.

[2] Y. Terada and S. Murata. Automatic assembly system for a large-scale modular structure - hardware design of module and assembler robot. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2349–2355 vol.3, Sep. 2004.

[3] Anis Koubaa. *Robot Operating System (ROS)*. 2016.

[4] Zhiyang Xiang, Zhu Xiao, Yourong Huang, Dong Wang, Bin Fu, and Wenjie Chen. Unsupervised and semi-supervised dimensionality reduction with self-organizing incremental neural network and graph similarity con-

straints. In James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 191–202, Cham, 2016. Springer International Publishing.

[5] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, USA, 2012. Curran Associates Inc.

[6] pybullet: Bullet real-time physics simulation, available at https://pybullet.org.

[7] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–387–I–395. JMLR.org, 2014.

[8] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2019.05.05].

[9] Fuchang Gao and Lixing Han. Implementing the nelder-mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277, Jan 2012.